Symulacja w Badaniach i Rozwoju
Vol. 2, No. 4/2011

Dariusz PIERZCHAŁA, Krzysztof CHLEBICKI
Wojskowa Akademia Techniczna, Wydział Cybernetyki,
ul. Gen. Sylwestra Kaliskiego 2, 00-908 Warszawa
E-mail: dariusz.pierzchala@wat.edu.pl, kchlebicki@wat.edu.pl

Metoda zwiększania przeżywalności federacji symulacyjnej

1 Wprowadzenie

W rozproszonym eksperymencie symulacyjnym opartym realizowanym w architekturze HLA (ang. High Level Architecture) może brać udział wiele autonomicznych symulatorów. Taki zestaw programów symulacyjnych pracujących pod nadzorem warstwy pośredniczącej RTI (ang. Runtime Infrastructure) nazywany jest federacją. Współpraca symulatorów sprowadza się do wymiany informacji o zróżnicowanej strukturze, często z dużą intensywnością. Cechą charakterystyczną komunikacji między programami (federatami) w federacji symulacyjnej zgodnej z architekturą HLA jest to, iż nie posiadają wiedzy o istnieniu ani położeniu pozostałych aplikacji, a cały ruch i wymiana danych odbywa się za pomocą wspomnianej warstwy programowej RTI. Dekompozycja systemów symulacyjnych wpłynęła bez wątpienia na możliwości, ale również wydajność i niezawodność całego środowiska eksperymentalnego. Problemy, które dotykają pojedyncze programy komputerowe (jak awarie sprzętu komputerowego i oprogramowania), nie omijają również symulatorów wchodzących w skład federacji. Wraz ze wzrostem liczby federatów rośnie ryzyko wcześniejszego i nieplanowego zakończenia pracy federacji, spowodowanego problemami z jednym lub wieloma jego składowymi. Dzieje się tak na przykład, gdy jeden z symulatorów do poprawnego działania wymaga danych, które dostarczyć może inny, a ten z chwilą awarii przestaje je publikować.

Błędy mogą występować w różnych składowych eksperymentu, a uwzględniając miejsca wystąpienia problemu można podjąć odrębne kroki mające na celu jego usunięcie. Na rysunku Rys. 1. przestawiono, w pewnym uogólnieniu, miejsca podatne na uszkodzenia.

Warstwa komunikacyjna niesie ze sobą problemy związane z uszkodzeniami mechanicznymi sprzętu: routery, kable. Sprzęt komputerowy, podobnie jak elementy warstwy sieciowej, również ulega awarii – przeciwdziałanie bądź naprawa sprowadza się w najprostszym scenariuszu do wymiany wadliwych komponentów. Awarie oprogramowania dotyczą warstwy systemu operacyjnego (OS), komponentów RTI bądź aplikacji federata. Błędnie działające biblioteki, wyjątki w pracy aplikacji lub systemu operacyjnego to częste sytuacje, gdy zawodzi oprogramowanie. O ile przygotowując środowisko do przeprowadzenia eksperymentu symulacyjnego mamy możliwość wyboru systemu operacyjnego lub komponentów RTI, to na tym przeciwdziałanie ewentualnym błędom się kończy. Tworząc program federata możemy przewidzieć problemy i zabezpieczyć się przez implementację obsługi wyjątków. Jednakże, jeśli w eksperymencie biorą udział symulatory różnego autorstwa, nie ma pewności, czy ich wytwórcy również przewidzieli podobne sytuacje i zapewnili obsługę na wypadek ich

wystąpienia. Warstwa federacji narażona jest na awarię głównie z powodu wadliwej pracy elementów składowych – federatów. W tej sytuacji wykonanie całego eksperymentu jest zagrożone, a sama federacja nie jest przed tym zabezpieczona. Ostatnim elementem w tej strukturze jest użytkownik. Problemy, jakich źródłem może być człowiek, są często nieprzewidywalne i jednocześnie istotne, gdyż posiada on dostęp oraz wpływa na każdą omawianą składową eksperymentu.

Rys. 1. Składowe eksperymentu podatne na błędy
Fig. 1. Components of an experiment error-prone

2 Błąd w ujęciu HLA

Błąd w ujęcie HLA jest to problem, który występuje w federacji lub środowisku, na którym uruchomiono eksperyment, a który uniemożliwia dalszą poprawną pracę federacji. Wyróżniamy dwa podstawowe typy błędów:

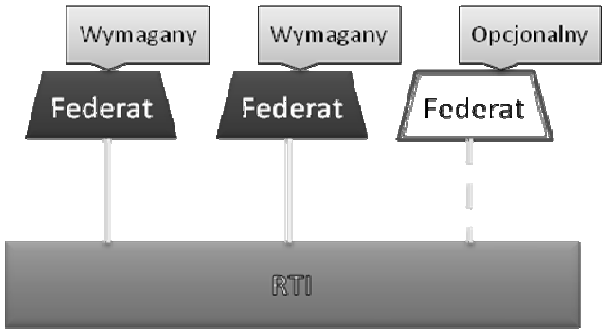
- *Utrata federata*, kiedy federacja traci jednego bądź wielu federatów, co skutkuje zagrożeniem przerwania całego eksperymentu. Informacja o utracie jednego z elementów jest rozgłaszana wewnątrz federacji zgodnie z ustalonymi i przyjętymi zasadami;
- *Utrata połączenia*, występuje kiedy federat traci połączenie z federacją. Federat przechodzi wówczas w stan „Not Connected" i podejmuje jak najszybciej próbę ustanowienia połączenia;

W przypadku wystąpienia jednego z opisanych błędów możliwe jest podjęcie różnych kroków, które pozwolą obsłużyć wyjątek zagrażający wykonaniu całego eksperymentu. W literaturze można spotkać podejścia, które sprowadzają się do [1,2,3,6,7]:

- *Niewidzialnej obsługi błędu* – podejście to zakłada automatyczne wykrycie i obsługę problemu, na przykład w komunikacji można zastosować inną metodę transportową;

- *Emulowania „utraconego" federata* – zastąpienie federata innym, który staje się aktywny w momencie jego zniknięcia z federacji. Jest to podejście najbardziej pożądane, jednak najtrudniejsze w implementacji;
- *Zastosowania federata „widmo"* – obok pierwotnej aplikacji federata istnieje jego działająca kopia. Jej stan odpowiada najbardziej aktualnemu stanowi aplikacji pierwotnej. Odzyskanie sprawności przez uszkodzonego federata wymaga ponownej synchronizacji ich stanów;
- *Automatycznego odłączenie federata* – w przypadku problemu z federatem, RTI odłącza go od federacji oraz restartuje operacje wymagające synchronizacji;
- *Zastosowania hierarchicznej federacji* – w ramach federacji występuje pewna grupa federatów (w tym wypadku pod-federacja), które są replikami. Jeśli federat uległ uszkodzeniu jego miejsce zajmuje kolejny, a jednocześnie z punktu widzenia eksperymentu sytuacja i zmiany są niewidoczne.

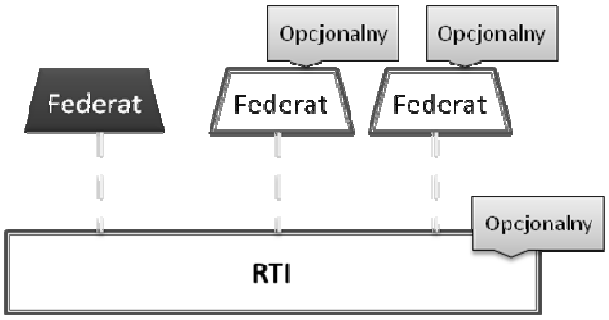
Osobnym zagadnieniem związanym z obsługą błędów wewnątrz federacji jest tolerowanie ich występowania. Możliwe jest takie zaprojektowanie eksperymentu oraz środowiska, na którym jest przeprowadzany, aby pewne problemy były przewidywalne podczas eksperymentu. W jednym z podejść określa się podzbiór wymaganych federatów, które mogą kontynuować eksperyment pomimo utraty jednego lub kilku innych federatów (Rys. 2).



Rys. 2. Federaci wymagani i opcjonalni

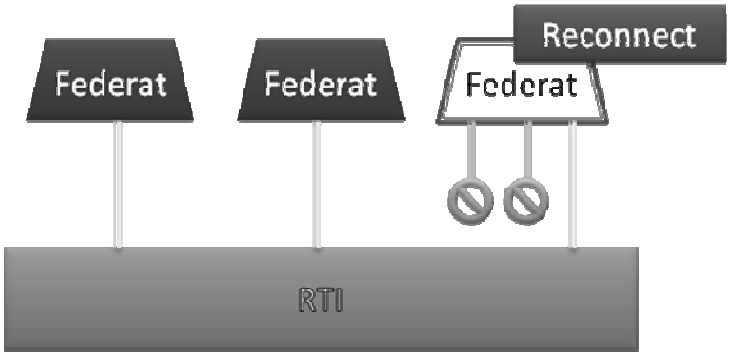
Fig. 2. Required and optional federates

W kolejnym podejściu opcjonalna jest federacja. Choć takie założenie odbiega od zasad przyjętego standardu HLA, w którym symulatory powinny współpracować ze sobą, to jednak okresowo podgrupa federatów może działać bez dostępu do federacji (warstwy RTI oraz co za tym idzie – innych federatów) i może wykonywać pewne aktywności poza kontrolą RTI.



Rys. 3. Opcjonalna federacja i okresowo autonomiczni federaci
Fig. 3. Optional federation and temporary autonomous federates

Podejściem kolejnym jest koncepcja ponownego pojawiania się federatów – zakłada ona, że federaci, którzy utracili połączenie z federacją, będą starali się powrócić tak szybko jak to możliwe. W tej sytuacji pozostałe symulatory będą oczekiwać na odłączonego federata, jeśli takie zdarzenie było przewidziane oraz obsłużone przez projektanta. Liczba wymaganych do przewidzenia zdarzeń dowodzi pracochłonności oraz złożoności procesu przygotowania i projektowania eksperymentu.



Rys. 4. Federat ponownie dołączany do RTI i federacji
Fig. 4. Federat re-attached to RTI and federation

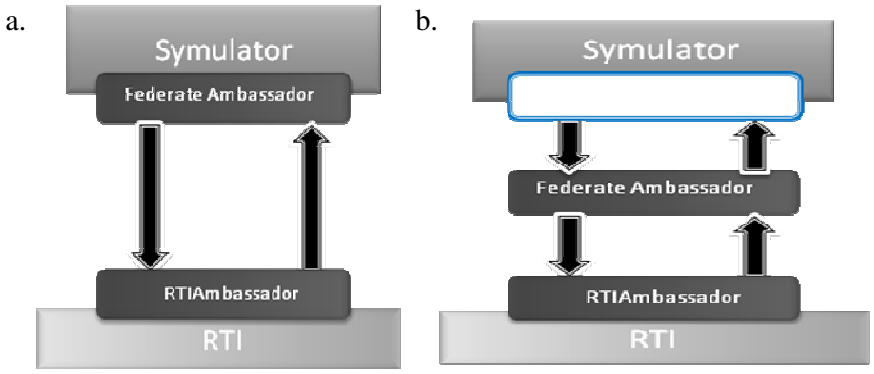
Rozwiązaniem bardziej rozbudowanym w porównaniu do przedstawionych powyżej jest zastosowanie specjalizowanego federata monitorującego. Jego zadaniem byłaby identyfikacja problemów w federacji oraz podejmowanie wybranych kroków z tych, które zostały przewidziane dla zaistniałej sytuacji. To podejście może łączyć wiele różnych rozwiązań, np. takich jak stosowanie duplikatów federatów. Możliwe jest użycie narzędzi monitorujących i zarządzających zasobami federacji (federatami) oraz pozwalających na zdalne uruchamianie i wznawianie pracy federata na innej maszynie niż pierwotnie federat działał [3]. W sytuacji uszkodzenia komputera lub grupy maszyn, na których uruchomiono symulator, wszystkie pliki oraz dane, włączając w to również aplikację, zostają przeniesione na wolny i sprawny węzeł sieci i tam ponownie uruchomione.

W trakcie uruchamiania kopii symulatora jego stan, czyli stan symulowanych obiektów, jest odtwarzany z kopii zapasowej tworzonej przyrostowo w trakcie trwania eksperymentu. Tworzeniem kopii zajmuje się również federat monitorujący.

Zauważalną wadą standardu architektury HLA jest brak pełnego wsparcia standardu dla obsługi błędów. W niniejszym opracowaniu zaproponowane zostało jedno z rozwiązań poprawiających przeżywalność federacji symulacyjnej.

3 Proponowane rozwiązanie - "proxy-federat"

W klasycznym eksperymencie symulacyjnym w architekturze HLA, symulatory (federaci) komunikują się z warstwą pośredniczącą RTI. Jednak nie odbywa się to na zasadzie bezpośredniego dostępu jednego komponentu do innego. Wymiana danych i komunikatów odbywa się przy zastosowaniu specjalizowanych interfejsów, zwanych ambasadorami. Na rysunku Rys. 5a. przedstawiony jest schemat typowej wymiany danych pomiędzy federatem oraz RTI.



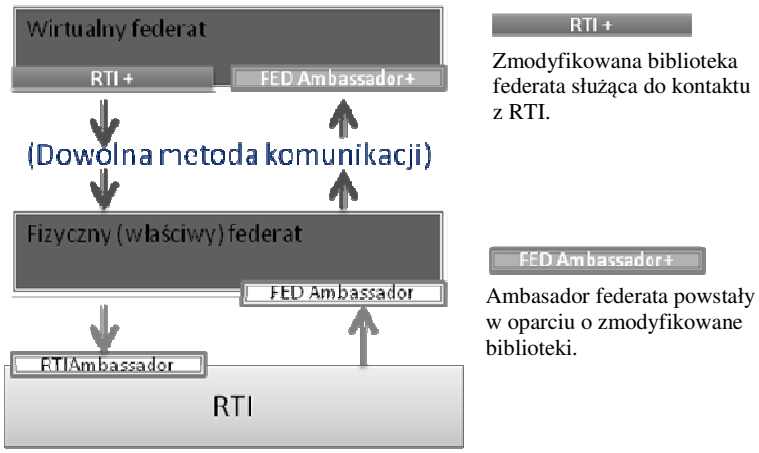
Rys. 5. a) Połączenie federat – RTI, b) Rozdzielenie interfejsu i federata
Fig. 5. a) Connection federate-RTI, b) Separation of interface and federate

Federat chcący skontaktować się z RTI wywołuje odpowiednią metodę jego interfejsu - "RTIAmbassador". Komunikacja zwrotna odbywa się w sposób analogiczny. Koncepcja proponowanego rozwiązania zakłada odseparowanie symulatora oraz jego ambasadora (interfejsu) (Rys. 5.b.) , co niesie ze sobą wiele korzyści - przede wszystkim może uchronić całą federację przed przedwczesnym zakończeniem działania, spowodowanym nieobsłużonym i krytycznym błędem w aplikacji federata.

Dokładniejszy diagram komunikacji w proponowanym rozwiązaniu znajduje się na rysunku Rys. 7. W zmodyfikowanej wg powyższej koncepcji strukturze wyróżnić należy dwie kluczowe składowe:

- Wirtualny federat – aplikacja symulatora, która w tym podejściu nie jest bezpośrednio połączona ze środowiskiem symulacyjnym oraz, co najważniejsze, nie jest już federatem w typowym rozumieniu HLA. Zamaskowanie tego przed samym symulatorem jest kluczowym założeniem rozwiązania;

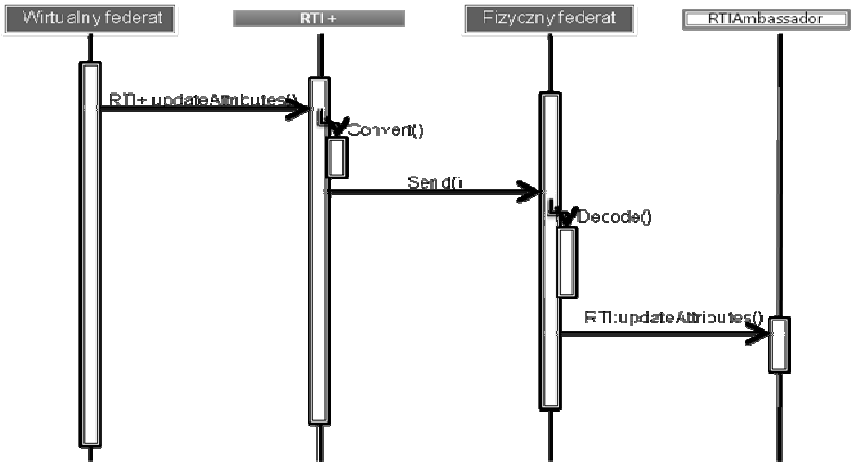
- Fizyczny federat (właściwy w sensie HLA)– sztucznie wprowadzona aplikacja, pełniąca rolę pośrednika, która ma na celu zapewnienie sprawnej i bezawaryjnej komunikacji między RTI oraz symulatorem. To właśnie jest wymieniany w tytule rozdziału "*Proxy-federat*".



Rys. 6. Współpraca wirtualnego i rzeczywistego federata
Fig. 6. Cooperation between virtual and real federates

Z poziomu eksperymentu oraz RTI nie ma różnicy, czy komunikacja odbywa się z federatem *proxy*, czy z właściwą aplikacją. Dopóki dane trafiają do symulatora oraz federat ma możliwość publikacji obiektów i interakcji, taki stan rzeczy świadczy o jego poprawnym działaniu. Programując aplikację federata korzysta się z wybranej implementacji HLA, framework'a pozwalającego na podłączenie do federacji symulacyjnej i współpracy w ramach eksperymentu. Separacja federata będzie możliwa dopiero po modyfikacji bibliotek wybranego pakietu. Z poziomu symulatora komunikacja poprzez mechanizmy pośredniczące jest niewidoczna. Zamiast komunikować się z RTI, federat wysyła żądanie do "*proxy-federata*", w tym celu zmodyfikować należy biblioteki służące do komunikacji z RTI. Sposób transportu na tym odcinku jest dowolny, w trakcie rozważań nad rozwiązaniem wybrano usługę web-serwis'ów jako uniwersalną, wydajną i prostą w użyciu metodę komunikacji. Właściwy federat po otrzymaniu żądania od federata wirtualnego dekoduje dane oraz wywołuje właściwą metodę interfejsu RTI - "RTIAmbassador". Na rysunku Rys. 7. zaprezentowano prostą sekwencję komunikacji w ramach działania symulatora. W trakcie zwykłej pracy, fizyczny federat jest również odpowiedzialny za wykonywanie kopii zapasowej stanu wszystkich symulowanych encji (obiektów). Nakłada to na niego wymóg analizy przesyłanych danych pod kątem zawartości oraz przeznaczenia. Pozwoli to na późniejszą obsługę procesu odtwarzania stanu federata. Proponowane rozwiązanie zakłada wykorzystanie punktów kontrolnych: „*checkpoint*". W przypadku powrotu uszkodzonego federata do poprawnej pracy, jego

stan zostaje odtworzony z wykorzystaniem ostatniego, najaktualniejszego punktu kontrolnego, co wiąże się z wykorzystaniem metod *rollback'u*.



Rys.7. Diagram sekwencji w komunikacji między federatem wirtualnym a RTI
Fig. 7. Sequence diagram of communication between virtual federate and RTI

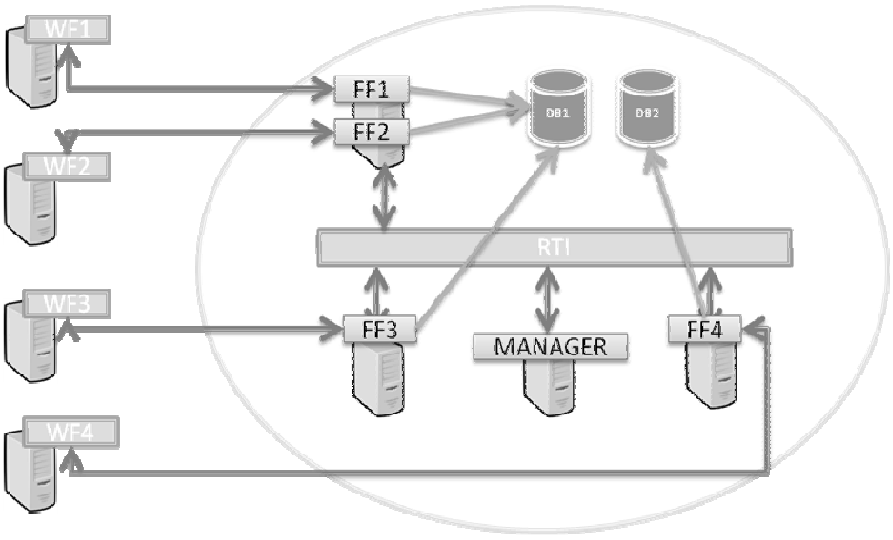
Kwestia wpływu mechanizmu odtwarzania stanu federata z błędnym dziaąniem na poprawność działania całej federacji (i pozostałych federatów) nie została jeszcze poddana badania na tyle dokładnie, aby możliwe było wskazanie rozwiązania najlepiej pasującego do omawianej koncepcji. Sam mechanizm składowania kopii zapasowych w formie punktów kontrolnych jest zagadnieniem szeroko rozpoznanym i posiadającym bogatą literaturę – wykorzystywany jest nie tylko w pracach z zakresu rozproszonych eksperymentów symulacyjnych.

Wirtualny oraz fizyczny federat nie muszą być uruchamiane na tym samym komputerze. Pozwala to na uniknięcie problemów związanych z błędami konkretnego zestawu sprzętowego. Uszkodzenie maszyny z działającym symulatorem bądź federatem *proxy* (ale nie jednoczesne) pozwoli ukryć fakt awarii przed federacją, RTI oraz innymi symulatorami. Jeśli elementem, który zawiódł będzie symulator, wówczas federat pośredniczący – wstrzyma eksperyment w federacji do czasu powrotu federata do stanu sprzed awarii. Jeśli to jednak "*proxy-federat*" zawiedzie, wówczas muszą zostać podjęte inne kroki, aby umożliwić dalszą poprawną realizację eksperymentu symulacyjnego. Dysponując określonymi zasobami na użytek eksperymentu (komputery, serwery baz danych, serwery aplikacyjne) można nimi zarządzać, podobnie do metody z wcześniejszego rozdziału. Jednak w omawianej koncepcji ta przenaszalność ograniczać się będzie do wyznaczenia miejsca uruchamiania federatów pośredniczących oraz miejsca składowania przez nich danych. Do sprawnej kontroli przebiegu tej czynności należy wprowadzić aplikację nadzorującą, której podstawowymi zadaniami będą:

- nadzór nad działaniem fizycznych federatów oraz procesem składowania danych;
- zarządzanie obciążeniem maszyn;
- monitorowanie przebiegu eksperymentu i zbieranie statystyk;

- wstrzymanie eksperymentu na czas odtwarzania stanu uszkodzonego symulatora.

Aplikacja pracować będzie jak zwykły federat, uczestnicząc w eksperymencie opartym na architekturze zgodnej z HLA, co zapewni typową i stosunkowo prostą wymianę komunikatów z federatami pośredniczącymi. Komunikacja z nimi jest bardzo istotna i ma na celu identyfikowanie uszkodzonych wirtualnych federatów (czyli najczęściej symulatorów). Pozwala również wykryć sytuację, gdy to "*proxy-federat*" jest tym elementem, który uległ awarii, oraz umożliwia stosowną reakcję w takiej sytuacji. Zadaniem federata nadzorującego, w momencie awarii pośrednika, jest wybór miejsca, w którym można uruchomić jego replikę. Na rysunku Rys. 8. przedstawiono przykład rozmieszczenia komponentów proponowanego rozwiązania.



Rys. 8. Architektura oparta na serwerze backupu oraz federacie nadzorującym
Fig. 8. Architecture based on backup server and controlling federate

Wirtualni federaci (WF1,2,3,4) wykonują się na oddzielnych zestawach sprzętowych. Federaci pośredniczący (FF1,2,3,4) zostają uruchomieni przez federata nadzorującego na wolnych węzłach sprzętowych struktury. Warto zaznaczyć, że nic nie stoi na przeszkodzie, aby działali na tych samych maszynach co symulatory. Jednak ze względu na możliwość awarii jednego komputera z uruchomionymi jednocześnie: wirtualnym i fizycznym federatem, taki przypadek jest niewskazany, gdyż oczekiwane zwiększenie przeżywalności federacji mogłoby nie zostać osiągnięte.

4 Podsumowanie

Wiele współczesnych systemów symulacyjnych wymaga aby złożony problem poddać procesowi dekompozycji i umożliwić uruchomienie w rozproszonym środowisku, gdzie pojęcie reużywalności jego komponentów i szybkości działania jest dobrze znane. Dlatego w dziedzinie symulacji powstał standard HLA, który jest obecnie ceniony za zapewnienie spójnej i wydajnej pracy pomiędzy symulatorami. Aby zrekompensować

braki architektury HLA w zakresie obsługi błędów, zaproponowana została własna metoda zwiększająca przeżywalność federacji symulacyjnej. Dzięki temu rozwiązaniu możliwe będzie przeprowadzenie eksperymentu w kontrolowanym środowisku, w którym błąd federata nie powinien zostać niezauważony i pominięty. W dalszym etapie należy przeprowadzić badania nad technikami, pozwalającymi uniknąć sytuacji, gdy po użyciu "*proxy-federata*" uzyskuje się skutek odwrotny do zakładanego: federacja staje się mniej stabilna niż przed jego wdrożeniem. Jako że zaproponowane rozwiązanie będzie funkcjonować jako element federacji w standardzie HLA, narażone jest na identyczne problemy jak te, przed którymi stara się zabezpieczyć innych federatów. Istotne wydaje się użycie techniki replikacji federatów pośredniczących, jak i zasadnym wydaje się posiadanie niezawodnego federata nadzorującego, potrafiącego szybko wykryć błąd i uchronić przed nim federację. Użycie mechanizmów w jakiejkolwiek federacji będzie wiązało się z jej modyfikacją oraz, bez wątpienia, wpłynie na pozostałe symulatory (federatów). Należy zatem ustalić, jaki narzut na wydajność eksperymentu będzie udziałem "*proxy-federata*" i jak duży wpływ na pozostałych uczestników eksperymentu będzie wymagany.

Literatura

1. Kiesling T.: *Fault-Tolerant Distributed Simulation: A Position Paper*. Institut für technische Informatik - Universität der Bundeswehr München 2005.
2. Moller B., Karlsson M., Lofstrand B.: *Developing Fault Tolerant Federation using HLA Evolved*. Pitch Technologies Nygatan.
3. Eklof M.,Moradi F., Ayani R.: *A framework for fault-tolerance in HLA-based distributed simulations*. Royal Institute of Technology Sweden 2006.
4. IEEE Standard 1516-2010: *IEEE standard for modeling and simulation (M&S) high level architecture (HLA) - framework and rules*, 2010.
5. Zengxiang L., Wentong C., Turner S., Ke P.: *Federate Fault Tolerance in HLA-based Simulation.* Technological University Singapore.
6. Damani O., Garg V.: *Fault-Tolerant Distributed Simulation*. University of Texas at Austin 2006.
7. Zengxiang L., Wentong C., Turner S., Ke P.: *A Replication Structure for Efficient and Fault-Tolerant Parallel and Distributed Simulations*. Technological University Singapore 2010.
8. Luthi J., Großmann S.: *A Flexible Framework for Fault Tolerant HLA Federations*. University of Applied Sciences FHS Kufstein Tirol 2004.
9. Stelling P., Foster I., Kesselman C., Lee C., Laszkowski G.: *A Fault Detection Service for Wide Area Distributed Computations*. University of South California.
10. Möller B., Morse K., Lightner M., Little R., Lutz B.: *HLA Evolved - A Summary of Major Technical Improvements*. 2007.
11. Eklöf M.: *Fault-Tolerance in HLA-Based Distributed Simulations*. 2006.
12. Pierzchała D.: Metoda wymiany informacji między heterogenicznymi systemami symulacji oraz wspomagania decyzji w systemach reagowania kryzysowego. *Symulacja w badaniach i rozwoju*, red. Leon BOBROWSKI, Vol. 1 No. 2/2010, 2010.

Streszczenie

W pracy omówiono metodę wykrywania i obsługi błędów w federacji symulacyjnej oraz zwiększania przeżywalności eksperymentu polegającą na rozdzieleniu aplikacji symulatora od interfejsu komunikacyjnego służącego do pracy pod kontrolą RTI. Zaproponowane podejście pozwala na ukrycie awarii aplikacji, wstrzymanie wszystkich działań w ramach federacji do momentu odzyskania sprawności przez uszkodzonego federata, bez przerywania całego eksperymentu. Wydzielony interfejs komunikacyjny pracuje podczas awarii, reaguje na informacje napływające z RTI oraz odtwarza stan federata do ostatniego zapamiętanego. W czasie poprawnej pracy aplikacji symulatora wszystkie aktualizacje symulowanych obiektów oraz wymieniane komunikaty zostają zapisane, aby mogły być następnie użyte w procesie odtwarzania.

A method for improvement fault-tolerance of a simulation federation

Summary

The paper presents the method for detecting and handling errors in the simulation federation and especially to improve fault-tolerance of distributed experiment. It can be obtained using the separation of application from the communication interface simulator. The proposed approach allows to hide the application failures, the suspension of all activities within the federation until the recovery efficiency of the faulty federate, without interrupting the experiment. Dedicated communication interface operates during the failure, responding to information coming from the RTI. Moreover, it restores state of the federate to the last stored. During proper work of the simulator all updates of simulated objects and the messages exchanged are stored in order to be used during restoring process (roll-back).